Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 210–229

Reconstruction of score sets

Antal IVÁNYI
Eötvös Loránd University,
Faculty of Informatics, 1117 Budapest,
Pázmány Péter sétány 1/C., Hungary
email: tony@inf.elte.hu

Abstract. The *score set* of a tournament is defined as the set of its different outdegrees. In 1978 Reid [15] published the conjecture that for any set of nonnegative integers D there exists a tournament T whose degree set is D. Reid proved the conjecture for tournaments containing $n = 1,\ 2$, and 3 vertices. In 1986 Hager [4] published a constructive proof of the conjecture for $n = 4$ and 5 vertices. In 1989 Yao [18] presented an arithmetical proof of the conjecture, but general polynomial construction algorithm is not known. In [6] we described polynomial time algorithms which reconstruct the score sets containing only elements less than 7. In [5] we improved this bound to 9.

In this paper we present and analyze new algorithms Hole-Map, Hole-Pairs, Hole-Max, Hole-Shift, Fill-All, Prefix-Deletion, and using them improve the above bound to 12, giving a constructive partial proof of Reid's conjecture.

1 Introduction

We will use the following definitions [3]. A *graph* $G(V, E)$ consists of two finite sets V and E, where the elements of V are called *vertices*, the elements of E are called *edges* and each edge has a set of one or two vertices associated to it, which are called its *endpoints* (*head and tail*). An edge is said to *join* its endpoints. A *simple graph* is a graph that has no self-loops and multi-edges.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C20, 68R10
Key words and phrases: score set, score sequence, approximation algorithms, Reid conjecture
DOI:10.2478/ausi-2014-0027

A *directed edge* is said to be *directed* from its *tail* and *directed* to its *head*. (The tail and head of a directed self-loop is the same vertex.)

A *directed graph* (*shortly: digraph*) is a graph whose edges are directed. If in a directed graph $(u, v) \in E$, then we say that u *dominates* v. An *oriented graph* is a digraph obtained by choosing an orientation (direction) for each edge of a simple graph. A *tournament* is a complete oriented graph. That is, it has no self-loops, and between every pair of vertices, there is exactly one edge. Beside the terms of graph theory we will use the popular terms player, score sequence, score set, point, win, loss etc.

A directed graph (so a tournament too) $F = (E, V)$ is *transitive*, if $(u, v) \in E$ and $(v, w) \in E$ imply $(u, w) \in E$.

The *order of a tournament* T is the number of vertices in T. A tournament of order n will be called an *n-tournament*.

An *(a, b, n)-tournament* is a loopless directed graph, in which every pair of distinct vertices is connected with at least a and at most $b \geq a$ edges. An (a, b, n)-tournament is *complete*, if in the matches any integer partition of c points is permitted for $a \leq c \leq b$.

The *score* (or *out-degree*) of a vertex v in a tournament T is the number of vertices that v dominates. It is denoted by $d_T^+(v)$ (shortly: $d(v)$).

The *degree sequence* (*score sequence*) of an n-tournament T is the ordered n-tuple $s_1, s_2, \ldots, s_n$, where s_i is the score of the vertex v_i, $1 \leq i \leq n$, and

$$s_1 \leq s_2 \leq \cdots \leq s_n. \tag{1}$$

An *n-regular sequence* is an increasingly ordered n-length integer sequence, that is an n-length score sequence is and n-regular sequence.

The *score set* of an n-tournament T is the ordered m-tuple $D = (d_1, d_2, \ldots, d_m)$ of the different scores of T, where

$$d_1 < d_2 < \cdots < d_m. \tag{2}$$

Figure 1 shows a $(1, 1, 4)$-tournament with score sequence $S = 0,\ 2,\ 2,\ 2$ and score set $D = \{0,\ 2\}$.

Theorem Landau [8, 10, 16] allows to test potential score sequences in linear time.

Theorem 1 (Landau [10]) *A nondecreasing sequence of nonnegative integers $S = s_1, s_2, \ldots, s_n$ is a score sequence of an n-tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \frac{k(k-1)}{2}, \quad 1 \leq k \leq n, \tag{3}$$

with equality when $k = n$.

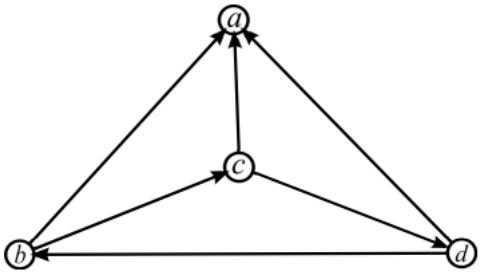

Figure 1: A tournament with score sequence 0, 2, 2, 2 and score set $\{0, 2\}$

Proof. See [10, 16]. □

To reconstruct a prescribed score set is much harder problem, then computing the score set belonging to the score sequence of a given tournament. Therefore surprising is the following conjecture published by Reid in 1978 [15]: if $m \geq 1$, $D = \{d_1, d_2, \ldots, d_m\}$ is a set of nonnegative integers, then there exists a tournament whose score set is D.

In his paper Reid described the proof of his conjecture for score sets containing 1, 2, and 3 elements, further for score sets representing an arithmetical or geometrical series. In 1986 Hager [4] published a constructive polynomial proof of the conjecture for $m = 4$ and $m = 5$.

In 2006 Pirzada and Naikoo gave a constructive proof of a special case of Theorem 3.

Theorem 2 (Pirzada and Naikoo [14]) *If $s_1, s_2, \ldots, s_m$ are nonnegative integers with $s_1 < s_2 < \cdots < s_m$, then there exists such $n \geq m$ for which there exists an n-tournament T with score set*

$$D = \left\{ d_1 = s_1,\ d_2 = \sum_{i=1}^{2} s_i,\ \ldots,\ d_m = \sum_{i=1}^{m} s_i \right\}. \tag{4}$$

Proof. See [14]. □

In 1976 Chartrand, Lesniak and Roberts [1] proved that for any finite set S of nonnegative integers there exists an oriented graph whose score set is S.

In 1989 Yao proved the conjecture of Reid.

Theorem 3 (Yao [18]) *If $m \geq 1$, $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ is a set of nonnegative integers, then there exists a tournament T with score sequence $S = s_1, s_2, \ldots, s_n$ such, that the score set of T is D.*

Proof. See [18]. □

The proof of Yao uses arithmetical tools and only proves the existence of the corresponding tournaments, but it does not give a construction.

In 1983 Wayland [17] proposed a sufficient condition for a set D of nonnegative integers to be the score set of a bipartite tournament. This result was improved to a sufficient and necessary condition in 1983 by Petrovič [12].

In [7] we proved that the extension of Yao's theorem is not true for k-tournaments (where every pair of vertices is connected with $k \geq 2$ edges).

Now we present three lemmas allowing a useful extension of Theorem 3.

Lemma 4 *If $d_1 \geq 1$, then the score set $D = \{d_1\}$ is realizable by the unique score sequence $S = d_1^{<2d_1+1>}$.*

Proof. If $|S| = n$ and S generates D then the sum of the elements of S equals to nd_1 and also to $n(n-1)/2$ implying $n = 2d_1+1$. Such tournament is realizable for example so, that any player P_i gathers one points against players $P_{i+1}, \ldots, P_{i+(n-1)/2}$ and zero against the remaining players (the indices are taken mod n). □

In this lemma and later $a^{<b>}$ means a multiset, in which a is repeated b times.The form of the score sequences using this notation is called *power form*.

Lemma 5 *If the score sequence $S = s_1, s_2, \ldots, s_n$ corresponds to the score set $D = \{d_1, d_2, \ldots, d_m\}$, then $n \geq d_m + 1$.*

Proof. If the score of a vertex v is d_m, then v dominates d_m different vertices. □

Lemma 6 *If $m \geq 2$ and the score sequence $S = s_1, s_2, \ldots, s_n$ corresponds to the score set $D = \{d_1, d_2, \ldots, d_m\}$, then*

$$2d_1 + 2 \leq n \leq 2d_m, \tag{5}$$

and both bounds are sharp.

Proof. Every element of D has to appear in S. Therefore the arithmetical mean of the scores is greater, than d_1, and smaller, than d_m. From the other side n-tournaments contain $B_n = \binom{n}{2}$ edges, so the arithmetical mean of their scores is $B_n/n = (n-1)/2$, therefore

$$d_1 < \frac{n-1}{2} < d_m, \tag{6}$$

implying (5).

For example if $k \geq 0$ and $D = \{k, k+1\}$, then according to (6) $n = 2k+2$ imply the sharpness of both bounds. □

The next extension of Theorem 3 is based on Lemmas 4, 5, 6.

Theorem 7 (Iványi, Lucz, Matuszka, Gombos [6]) *If $m \geq 1$ and $D = \{d_1, d_2, \ldots, d_m\}$ is an increasingly ordered set of nonnegative integers, then*

- *there exist a tournament T, whose score sequence is S and score set is D;*
- *if $m = 1$, then $S = s_1^{<2d_1+1>}$;*
- *if $m \geq 2$, then*

$$\max(d_m + 1, 2d_1 + 2) \leq n \leq 2d_m; \tag{7}$$

- *the bounds in (7) are sharp.*

Proof. The assertion follows from the above lemmas (see [6]). □

Taking into account the remark of Beineke and Eggleton [16, page 180] we can formulate Reid's conjecture as an arithmetical statement without the terms of the graph theory. Let $D = \{d_1, d_2, \ldots, d_m\}$ be an increasingly ordered set of nonnegative integers. According to the conjecture there exist positive integer exponents $x_1, x_2, \ldots, x_m$ such that

$$S = d_1^{<x_1>}, d_2^{<x_2>}, \ldots, d_m^{<x_m>} \tag{8}$$

is the score sequence of some $(\sum_{i=1}^{m} x_i)$-tournament. Using Landau's theorem it can be easily seen that Reid's conjecture is equivalent to the following statement [13, 18].

For every set $D = \{d_1, \ldots, d_m\}$ with the property $0 \leq d_1 < d_2 < \cdots < d_m$ there exist positive integers $x_1, \ldots, x_m$, such that

$$\sum_{i=1}^{k} x_i d_i \geq \binom{\sum_{i=1}^{k} x_i}{2}, \quad \text{for } k = 1, \ldots, m-1, \tag{9}$$

and

$$\sum_{i=1}^{m} x_i d_i = \binom{\sum_{i=1}^{m} x_i}{2}. \tag{10}$$

Commenting Yao's proof Qiao Li wrote in 1989 [11]: *Yao's proof is the first proof of the conjecture, but I do not think it is the last one. I hope a shorter and simpler new proof will be coming in the near future.*

However, the constructive proof has not been discovered yet.

Our algorithms investigate only the zerofree score sets, The base of this approach is the following lemma.

Lemma 8 *Let $m \geq 2$. A sequence $S = s_1^{<e_1>}, s_2^{<e_2>}, \ldots, s_n^{<e_m>}$ is the score sequence corresponding to the score set $D = \{0, d_2, d_3, \ldots, d_m\}$ if and only if the sequence $S' = (s_2 - 1)^{<e_2>}, (s_3 - 1)^{<e_3>}, \ldots, (s_n - 1)^{<e_n>}$ is the score sequence corresponding to $D' = \{d_2 - 1, d_3 - 1, \ldots, d_m - 1\}$.*

Proof. Part *if* of the proof: If S is the score sequence corresponding to D then $s_1 = 0$ and $e_1 = 1$ that is all other players won against the player having the score $s_1 = 0$, so S' corresponds to D'.

Part *only if* of the proof: If S' corresponds to D', then we add a new score $d_1 = 0$ to D', increase the multiplicity of the other scores by 1 and get D which correspond to S. □

2 Reconstruction of score sets of tournaments

Earlier [5, 6] we proposed polynomial approximate algorithms Balancing, Shortening, and Shiftening.

By computer experiments we proved that they reconstruct all score sets with $d_m < 9$. We also described exact brute force algorithms Sequencing and Diophantine [6].

The proposed algorithms reconstruct the majority of the score sets with $d_m = 9$. , but there are three exceptional sets with $d_m = 9$. Exceptions are the sets $\{2, 4, 5, 6, 7, 8, 9\}$, $\{1, 2, 5, 6, 7, 8, 9\}$, and $\{1, 2, 4, 7, 8, 9\}$.

In this paper we present new polynomial algorithms Hole-Pairs, Hole-Max, Hole-Shift, Prefix-Shift, Fill-All and using these theorems we improve the earlier bound to $d_m < 12$. Our algorithms are based on Theorem 7. Since there are quick (quadratic) algorithms to construct n-tournaments corresponding to a given score sequence, our algorithms construct only a suitable score sequence.

If the score sequence of a tournament is S and its score set is D, then we say, that S *generates* D, or D *corresponds* to S. If D is given, then we call the corresponding score sequence *good*.

3 Hole-type algorithms

The Hole-type algorithms are based on the idea that we take the transitive score sequence $s = 0,\ 1,\ \ldots,\ d_m$ and gradually remove from it the elements corresponding to the missing elements of the investigated score set.

In a score set $D = \{d_1, d_2, \ldots, d_m\}$ there is a *k-hole* before element d_i $(1 \leq i \leq m)$, if

- $d_1 = k + 1$, or
- $2 \leq i \leq m$ and $d_i - d_{i-1} = k$.

In the first case the hole is *outer*, while in the second case it is an *inner* hole. The missing elements are called *hole elements*, while the neighbors of a hole are its *lower*, resp. *upper* bound. During the maintenance of the score set D a hole is *active*, if the elements of the hole are present in s. At the beginning all holes are active. A hole is *passive*, if its elements are missing from the actual score sequence. During the reconstruction of D we gradually transform the active holes to passive. The reconstruction process is finished, when D contains only passive holes.

We prepare the work of the HOLE-type algorithms with the construction of a *hole-map*.

3.1 Algorithm HOLE-MAP

The *hole map* of a score set $D = \{d_1, \ldots, d_m\}$ is an $(m \times d_m)$-sized array $H(D) = H[1\ .\ .m, 1\ .\ .d_m]$, where the j-th column of $H(D)$ describes the j-sized active holes: $H[i, j]$ gives the beginning address of the i-th j-sized hole in D (if there exists such hole, otherwise $H[i, j]$ is undetermined).

The next algorithm HOLE-MAP generates the hole map $H(D)$, further the number of active holes $N[0](D)$ and the number of active i-holes $N[i](D)$ $(1 \leq i \leq d_m)$ in D. $N(D)$ is *the frequency vector* of the active holes.

The pseudocodes of this paper are written using the conventions proposed in the textbook [2].

Input. $m = m(D)$ $(m \geq 1)$: the number of the elements of D;
$D = \{d_1, \ldots, d_m\}$: a score set containing m elements.

Global variables. $D = \{d_1, \ldots, d_m\}$: score set;
m: the number of elements of D;
H: hole map of D.

Working variables. $i,\ j$: cycle variables.

Output: $H[1\ .\ .\ m, 1\ .\ .\ d_m]$: the hole map of D;
$N[0\ .\ .\ m] = N(D)$: the hole frequency vector of D.

HOLE-MAP(m, D)

```
01 for i = 0 to d_m                          // Line 01–03: initialization
02     N[i] = 0
03 if d_1 > 0                                // Line 03–06: is there an outer hole?
04     N[0] = 1
05     N[d_1] = 1
```

06 $H[1, d_1] = 0$
07 **for** $j = 1$ **to** $m - 1$ // Line 07–11: investigation of the inner holes
08 **if** $d_{j+1} - d_j > 1$
09 $N[d_{j+1} - d_j - 1] = N[d_{j+1} - d_j - 1] + 1$
10 $N[0] = N[0] + 1$
11 $H[N[d_{j+1} - d_j - 1], d_{j+1} - d_j - 1] = d_j + 1 + 1$
12 $H[N[0] = 0$
13 'no inner hole'
14 **return** H, N // Line 13: return of the results

If we use algorithm Hole-map for the score set $D = \{2, 4, 6, 7\}$, then the input is D and $m = 4$, the size of the vector N is 7. Table 1 contains $H(D)$.

beginning/hole size	1	2	3	4	5	6	7
first hole	2	0	−−	−−	−−	−−	−−
second hole	4	−−	−−	−−	−−	−−	−−
third hole	−−	−−	−−	−−	−−	−−	−−
fourth hole	−−	−−	−−	−−	−−	−−	−−

Table 1: Hole map $H(D)$ of the score set $D = \{2, 4, 6, 7\}$

The running time of Hole-map is $\Omega(d_m)$ in all cases.

3.2 Algorithm Hole-Pairs

Two i-sized holes form a *hole pair*. The hole which appear earlier in D is called *lower hole*, while the other one is called *upper hole*.

Our algorithms do not change the result of the matches between players in the active holes and any other player. Since for the initial transitive sequence is it true that the players are defeated by any other player having larger score, this property remains true (in such sense that there exists such tournament which realizes the given score sequence and has this property) for the players whose score is in an active hole after the application of Hole-Pairs and Hole-Max. We can call the such sequences *hole-transitive*.

The next algorithm Hole-Pairs forms the maximal possible number, that is $\lfloor N[i]/2 \rfloor$, of pairs of i-holes of D for $1 \leq i \leq d_m$ and removes the hole elements from the corresponding sequence.

Input. $N(D)$: the frequency vector of the active holes in D.

Global variables. $D = \{d_1, \ldots, d_m\}$: score set;
m: the number of elements of D;

H: hole map of D;
$s = 0, 1, \ldots, d_m$: the starting transitive score sequence;
Working variables. $i,\ j$: cycle variables.
Output. $t = t_0, t_1, \ldots, t_{d_m}$: the sequence produced by HOLE-PAIRS;
$M(D)$: the frequency vector of the active holes in D.

HOLE-PAIRS(N, s)

```
01 for i = 0 to d_m                                  // Line 01–03: initialization
02     M[i] = N[i]
03     t[i] = s[i]
04 for i = d_m downto 1                  // Line 04–11: processing of the hole pairs
05     while M[i] > 1
06         for j = H[M[i], i] to H[M[i], i] + i − 1
                                  // Line 06–07: maintenance of the upper hole
07             t[j] = t[H[M[i], i]] − 1
08         for j = H[M[i] − 1, i] to H[M[i] − 1, i] + i
                                  // Line 08–09: maintenance of the lower hole
09             t[j] = t[H[M[i − 1] + i, i]] + 1
10         M[i] = M[i] − 2                    // Line 10: updating of M[i]
11 return t, M                                // Line 11: return of the results
```

If the input of HOLE-PAIRS is the hole map $H(D)$ of the score set $D = \{2, 4, 6, 7\}$, then the algorithm starts with the transitive score sequence $t = 0, 1, \ldots, 7$, and processing the 1-holes at 3 and 5 gets the shortened sequence $t = 0, 1, 2, 4^3, 6, 7$. The number of active holes in D is reduced to $M[0] = 1$.

The running time of HOLE-PAIRS is $\Theta(d_m)$ in all cases.

3.3 Algorithm HOLE-MAX

If $M[0](D) > 0$, then we have at least one active hole and can simply eliminate the largest one: if the largest hole is a c-hole, then we add the elements $d_m + 1,\ d_m + 2,\ \ldots,\ d_m + c$ to t and reduction of these elements to d_m allows us to increase the elements of the c-hole to its upper bound, as the next algorithm HOLE-MAX makes.

Input. $M(D)$: the updated frequency vector of the active holes in D;
$t = t_0, t_1, \ldots, t_{d_m}$: the reduced sequence produced by HOLE-PAIRS.
Global variables. D: score set;
m: the number of elements of D;
H: hole map of D.
Working variables. $i,\ j,\ k$: cycle variables.

Output. $u = u_0, u_1, \ldots, u_{d_m+c}$: the reduced score sequence produced by HOLE-MAX;
$O(D)$: the updated frequency vector of the active holes in D;
c: the size of the largest active hole in t.

HOLE-MAX(M, t)

```
01 for i = 0 to d_m                          // Line 01–02: initialization
02     O[i] = M[i]
03 c = d_m                                   // Line 03–14: elimination of the largest hole
04 while M[c] = 0                            // Line 04–05: seeking of the largest hole
05         c = c − 1
06 for j = 0 to d_m                          // Line 06–09: initialization u
07     u_j = t_j
08 for j = d_m + 1 to d_m + c
09     u_j = j
10 for k = 1 to c                            // Line 10–12: maintenance of the lower hole
11     u_{H[1,c]+k} = u_{H[1,c]+c}
12     u_{d_m+k} = u_{d_m}
13 O[c] = O[c] − 1                           Line 13–14: updating of number of active holes
14 O[0] = O[0] − 1
15 return u, c, O                            Line 15: return of the result
```

If $D = \{2, 4, 6, 7\}$ and the input of algorithm HOLE-MAX is the sequence $t(D)$, then HOLE-MAX eliminates the remained hole and we get the score sequence $u = 2^3, 4^3, 6, 7^3$ corresponding to D.

Running time of HOLE-MAX is $\Omega(d_m)$ in all cases.

3.4 Algorithm HOLE-SHIFT

After algorithm HOLE-PAIRS there is at most one i-hole for every i, that is $0 \le O[i] \le 1$ for every $1 \le i \le d_m$. Algorithm HOLE-MAX eliminates the largest hole by appending c scores $d_m + 1, \ldots, d_m + c$ to the input sequence t and reducing these scores to d_m.

Let the player having u_j points be T_j $(1 \le j \le d_m + c)$. The following algorithm HOLE-SHIFT uses the power form $w = w_1^{<e_1>}, \ldots, w_q^{<e_q>}$ of the input sequence u. An element w_k of w is called *point-sender* if its exponent e_k is greater than 1. If w_j and w_k are upper bounds of active holes and $j < k$, then T_k can send $e_k - 1$ points—through the player T_j—to the players in the lower active hole to increase their scores to w_k.

The *hole list* is a pair of vectors consisting of the *begin vector* $b(D) = b_1, \ldots, b_{O[0]}$ and the *length vector* $l(D) = l_1, \ldots, l_{O[0]}$, where b contains the beginning indices of the active holes in increasing order, while l contains the length of the corresponding holes. E.g. if $D = \{2, 5, 9\}$, then $b(d) = 0, 3, 6$ and $l(D) = 2, 2, 3$.

Hole-Shift uses also the score sequence $w = w_1^{<e_1>} \ldots w_n^{<e_n>}$ which is the power form of u. If $2 \leq j \leq q$, $e_j > 1$, and $w_j - w_{j-1} = 1$, then $e_j - 1$ players from the players having w_j points can give one pointto any player having smaller index. These $e_j - 1$ points are called *free points*. The following algorithm Hole-Shift extends this idea for the case $w_j - w_{j-1} > 1$.

The next algorithm Hole-Shift finds the largest holes in the investigated score sequence and tries to remove this hole by shifting of the points from the point-sender scores to the scores in the given active hole.

Input. $O(D)$: $O[0]$ is the number of active holes in D, $O[i]$ $(1 \leq i \leq d_m)$ is the number of active i-holes in D;
V: the sum of the sizes of the active holes in D;
$u = u_0, u_1, \ldots, u_{d_m+c}$: the reduced sequence produced by Hole-Max.

Global variables. D: score set;
m: the number of elements of D;
H: hole map of D;
c: the length of largest hole removed by Hole-Shift.

Working variables. $b(D) = b_1, \ldots, , b_{O[0]}$: the begin vector of the active holes of D;
$l(D) = l_1, \ldots, l_{O[0]}$: the length vector of the active holes of D;
$w(D) = w_1^{<e_1>} \ldots w_q^{<e_q>}$: power form of u;
$b(D) = b_1, \ldots, b_{O[0]}$ q: the length of the power form of u;
g: size of the actual largest active hole;
a: the index of the investigated largest active hole;
$h = O[0]$: the number of active holes;
s: difference between two consecutive scores in w;
r: number of points required by the investigated active hole to shift scores in an active hole to theirs upper bound;
i, j, k: cycle variables.

Output. $v = v_1, \ldots, v_{d_m+c}$: the reduced score sequence produced by Hole-Shift;
$P(D)$: the updated frequency vector of D.

Hole-Shift(O, u)

01 **for** $i = 0$ **to** d_m // Line 01–02: initialization

```
02      P[i] = O[i]
03 for i = 1 to d_m + c
04      v[i] = u[i]
06 w_1 = u_1                  // Line 06–14: computation of the power form of u
07 e_1 = 1
08 q = 1
09 for k = 2 to d_m + c
10      if u_k = u_{k-1}
11         e_q = e_q + 1
12      else q = q + 1
13           w_q = u_k
14           e_q = 1
15 if d_1 > 0 and u_1 == 0          // Line 15–24: computation of b, l, and h
16    b_1 = 0
17    l_1 = d_1
18    h = 1
19 else h = 0
20       for i = 1 to m + c - 1
21           if d_{i+1} - d_i ≥ 2
22              h = h + 1
23              b_h = d_i + 1
24              l_h = d_{i+1} - d_i - 1
25 while P[0] > 0                          // Line 25: while is active hole
26         g = d_m              // Line 26–31: finding of the largest active hole
27         a = 1
28         for i = 1 to h
29             if l_i > l_a
30                a = i
31                g = l_a
32         r = g(g + 1)/2           // Line 32: number of the required points
33         while r > 0
34                j = 1              // Line 34–36: finding the starting score;
35                while w_j < b_a + l_a and e_j = 1 and w_j - w_{j-1} > r and j ≤ q
36                      j = j + 1
37                if j > q                                  // Line 37–38: defeat
38                   return                  // 'HOLE-SHIFT is not sufficient'
39                y = ⌊r/(w_j - w_{j-1})⌋
40                r = r - min(e_j - 1, y)(w_j - w_{j-1})
41                e_j = e_j - min(e_j - 1, y)
```

42 **for** $i = 1$ to l_a
43 $w_{j-1} + 1 = w_{j-1} + l_a$
44 $q = q - l_a$
45 **return** P, w, v

The running time of Hole-Shift is $O(md_m)$, since it is determined by the cycles in Lines 33–41, and there are at most m holes and for every hole at most d_m players can give a point.

3.5 Algorithm Fill-All

If $d_m < 10$, then our previous algorithms can reconstruct all possible score set, but if $d_m = 10$, then there are two critical sets, and if $d_m = 11$, then there are three ones. If $d_m = 10$, then the score sets $\{1, 3, 4, 5, 8, 10\}$ and $\{1, 2, 3, 5, 8, 10\}$, while if $d_m = 11$, then the score sets $\{2, 3, 4, 5, 6, 7, 8, 9, 11\}$, $\{1, 2, 3, 5, 7, 11\}$, and $\{1, 2, 3, 4, 5, 6, 7, 8, 11\}$ are unsolvable for them.

The following Fill-All reconstructs these score sets. The basic idea of Fill-All is that at first we add new elements to the starting sequences, sufficient to cover the point requirements of all holes. If we are lucky then the number the additional points is equal to the total point requirement. Otherwise we gradually increase the number of additional scores and try to hide the additional points.

Input. $Q(D)$: $Q[0]$ is the number of active holes in D, $Q[i]$ $(1 \leq i \leq d_m)$ is the number of active i-holes in D.

Global variables. D: score set;
m: the number of elements of D.

Working variables. T: the total point requirement of the holes;
a: the number of added new scores;
$p = a(a-1)2$: the number of additional points;
q: the number of free additional points;
$w(D) = w_1^{<e_1>} \dots w_q^{<e_q>}$: power form of u;
$e = e_1, \dots, e_q$: e_i $(1 \leq i \leq q)$ is the exponent od w_i;
i, j: cycle variables.

Output. $e = e_1, \dots, e_m$: the exponents of the score sequence corresponding to D.

Fill-All($m.D$)
01 $T = 0$ // Line 01–06: computation of the point requirement of the holes
02 **if** $d_1 > 0$
03 $T = d_1$

```
04 for i = 2 to m
05     if d_i − d_{i−1} > 1
06         T = T + (d_i − d_{i−1})(d_i − d_{i−1} − 1)/2
07 e_1 = 1                          // Line 07–15: computation of the exponents
08 if d_1 > 0
09    e_1 = d_1
10 for j = 2 to m − 1
11     e_j = 1
12     if d_j − d_{j−1} > 1
13         e_j = d_j − d_{j−1}
14 a = min(k | k(k − 1)/2 ≥ T)  // Line 14: first number of additional scores
15 for k = a to d_m     // Line 15–23: testing of the potential score sequences
16     p = k(k − 1)/2                  // Line 16: number of additional points
17     q = p − T                   // Line 17: number of free additional points
18     for i = 1 to m − 1
19         if e_{i+1} − e_i > 1 and d_{i+1} − d_i ≤ q
20             f = min(⌊q/(d_i − d_{i−1}⌋, e_{i−1})
21             e_{i−1} = e_{i−1} − f
22             e_i = e_i + f
23             q = q − f(d_i − d_{i−1})
24             if q == 0
25                 return e
26 print 'algorithms are not sufficient'          // Line 26–27: algorithms
27 stop                                          // could not reconstruct D
```

The running time of Fill-All is $O(d_m^2)$ in all cases.

4 Algorithm Prefix-Deletion

The earlier algorithms can not reconstruct the score sets $\{1, 7, 10\}$ and $\{1, 7, 11\}$.

Part of the earlier described algorithm Shortening [5, 6] is the deletion of the leading zero element from a score set, and decreasing of the remaining elements by 1.

Now we generalize this idea. If the score set $D = \{d_1, \ldots, d_m\}$ begins with 1, then according to the theorem of Landau a corresponding score sequence can contain one, two or three 1's. If it contains exactly three 1's, then each of the corresponding players gathered one point in their minitournament, therefore provisionally deleting them we get the smaller score set $D' = \{d_2 - 3, d_3 - 3, \ldots, d_m - 3\}$. In the concrete case, when $D = \{1, 4, 7\}$ Balancing results the solution $s(D') = 4^6, 7^6$, resulting $s(D) = 1^3, 7^6, 10^6$.

In general case we have to investigate also the cases when the corresponding sequence contains one or two 1's.

It is a natural idea to extend the investigation to the general case $1 \leq d_1 \leq k$, when according to Landau's theorem the corresponding sequences can start with $1 \leq p \leq 2d_1 + 1$ d_1's.

Since in the case $d_m < 12$ it is sufficient to consider the case $k = 1$ and $p = 3$, we present only pseudoprogram for this special case.

In the program we call the procedure SEQUENCE-BASE which handles the results of the previous reconstruction algorithms for all score sets with $d_m < 12$. If we wish to reconstruct a score set, whose largest element is d_m, then the corresponding data base has to contain 2^{d_m-1} score sequence, but the search in it requires only $O(m)$ time.

Its input is the reduced variant of D ($D' = \{d_2 - 3, d_3 - 3, \ldots, d_m - 3\}$), and the output is the score sequence corresponding to D' ($w = w_1, \ldots, w_y$), further its length (y).

Input. $Q(D)$: the updated hole frequency vector;
$v = v_0, v_1, \ldots, v_{d_m+c}$: the reduced sequence produced by HOLE-MAX.

Global variables. D: score set;
m: the number of elements of D;
H: hole map of D-

Working variables. $D' = \{d'_1, \ldots, d'_{m-1}\} = \{d_2, \ldots, d_m\}$: the shortened score set;
$w(D) = w_1^{<e_1>} \ldots w_q^{<e_q>}$: power form of u;
i, j: cycle variables.

Output. $x = x_1, \ldots, x_{y+3}$: the score sequence corresponding to D.

PREFIX-DELETION(Q, v)

```
01 for i = 1 to d_m                          // Line 01–05: initialization
02     R[i] = Q[i]
03     w[i] = v[i]
04 for i = d_m + 1 to d_m + c
05     w[i] = v[i]
06 if m < 4 or d_1 ≠ 1 or d_2 < 3            // Line 06–07: defeat
07    'PREFIX-DELETION is not sufficient'
08 for i = 1 to m − 1
09     d'_i = d_{i+1}
10 SEQUENCE-BASE(D')                         // Line 10: call of SEQUENCE-BASE
11 x_1 = x_2 = x_3 = 1 // Line 11: reconstructed sequence begins with three 1's
12 for j = 4 to y + 3 // Line 12–13: computation of the further elements of x
```

```
13      x_j = w_{j-3}
14 return x                                   // Line 14: return of the results
```

Running time of PREFIX-DELETION is $\Omega(d_m)$ in all cases.

5 The main program

The previous pseudocodes are procedures. These procedures are called by the following main program MAIN.

MAIN(m, D)

```
01 HOLE-MAP(m, D)                    // Line 01: construction of the hole map
02 if N[0] = 0                       // Line 02–04: printing the solution
03    print 'no hole in the score set'
04    stop
05 HOLE-PAIRS(N)                     // Line 05: construction of the hole pairs
06 if M[0] = 0                       // Line 06–08: printing the solution
07    print t
08    stop
09 HOLE-MAX(M, t)                    // Line 09: filling of the longest hole
10 if O[0] = 0                       // Line 10–12: printing the solution
11    print u
12    stop
13 HOLE-SHIFT(P, u)                  // Line 13: shifting of the holes
14 if P[0] = 0                       // Line 14–16: printing the solution
15    print v
16    stop
17 PREFIX-DELETION(Q, v)             // Line 17: shifting of the holes
18 if Q[0] = 0                       // Line 18–20: printing the solution
19    print w
20    stop
21 FILL-ALL(N, w)                    // Line 21: filling of the holes
22 if q = 0                          // Line 22–24: printing the solution
23    print e
24    stop
25 PREFIX-DELETION(Q, v)             // Line 25: deletion a prefix
26 if Q[0] = 0                       // Line 26–28: printing the solution
27    print w
28    stop
29 print 'the algorithms are not sufficient'   // Line 29–30: defeat
30 stop
```

In worst case the running time of MAIN is $O(d_m^2)$.

6 Simulation results

Algorithm BALANCING reconstructs all score sets with $d_m < 6$, but can not reconstruct the score sets $\{1, 3, 6\}$ and $\{1, 2, 3, 5, 6\}$ [6]. For these critical score sets algorithm SHORTENING gives the solutions $1^3, 3, 6^5$ and $1^2, 2, 3^2, 5, 6$. These algorithms can not reconstruct the score sets $1, 2, 3, 5, 7$ and $1, 2, 3, 4, 6, 7$. In the first case algorithm SHIFTENING results a corresponding score sequence $1^2, 2, 3^2, 5, 7^3$, while in the second case it gives $1, 2, 3, 4^2, 6^4, 7$ [5].

n	D	s	Algorithms	DIOPHANTINE
1	$\{2, 4, 5, 6, 7, 8, 9, 10\}$	$3, 2, 1, 1, 1, 1, 2, 2$	M + S	$4, 1, 1, 1, 1, 1, 2, 1$
2	$\{2, 3, 5, 7, 9, 10\}$	$3, 1, 2, 3, 2, 2$	P + M + S	$3, 2, 2, 2, 1, 1$
3	$\{1, 7, 10\}$	$3, 6, 6$	X	same
4	$\{1, 4, 5, 7, 9, 10\}$	$2, 3, 1, 3, 2, 2$	P + M + S	$3, 1, 3, 2, 1, 1$
5	$\{1, 3, 6, 8, 9, 10\}$	$2, 2, 4, 1, 2, 2$	P + M + S	$1, 5, 2, 1, 1, 1$
6	$\{1, 3, 6, 7, 8, 10\}$	$2, 2, 3, 2, 1, 3$	P + M + S	$3, 1, 4, 1, 1, 1$
7	$\{1, 3, 4, 7, 9, 10\}$	$2, 2, 1, 4, 2, 2$	P + M + S	$3, 1, 4, 1, 1, 1$
8	$\{1, 3, 4, 5, 8, 10\}$	$2, 2, 1, 1, 3, 5$	F	$2, 2, 2, 1, 3, 1$
9	$\{1, 2, 5, 7, 9, 10\}$	$2, 1, 3, 3, 2, 2$	P + M + S	$2, 1, 5, 1, 1, 1$
10	$\{1, 2, 5, 6, 7, 8, 10\}$	$2, 1, 3, 1, 1, 2, 3$	P + M	$2, 2, 2, 1, 1, 2, 1$
11	$\{1, 2, 4, 8, 9, 10\}$	$2, 2, 1, 4, 1, 4$	P + M	$1, 3, 2, 4, 1, 1$
12	$\{1, 2, 4, 6, 9, 10\}$	$2, 1, 2, 2, 3, 4$	P + M	$2, 1, 2, 4, 1, 1$
13	$\{1, 2, 3, 7, 8, 10\}$	$2, 1, 1, 4, 2, 4$	P + M	$1, 1, 4, 2, 2, 1$
14	$\{1, 2, 3, 5, 8, 10\}$	$2, 1, 1, 2, 3, 5$	F	$1, 2, 2, 2, 3, 1$
15	$\{1, 2, 3, 5, 8, 9, 10\}$	$2, 1, 2, 1, 3, 1, 3$	P + M	$1, 2, 1, 4, 1, 1, 1$
16	$\{1, 2, 3, 5, 7, 10\}$	$2, 1, 1, 3, 1, 6$	A	$2, 1, 1, 2, 4, 1$
17	$\{1, 2, 3, 5, 6, 9, 10\}$	$2, 1, 2, 1, 1, 3, 3$	P + M	$2, 1, 1, 1, 4, 1, 1$
18	$\{1, 2, 3, 4, 6, 7, 10\}$	$2, 1, 1, 2, 1, 1, 5$	P + M	$2, 1, 1, 1, 1, 4, 1$

Table 2: Reconstruction results of the critical score sets ending with $d_m = 10$

BALANCING, SHORTENING, and SHIFTENING reconstruct the majority of score sets with $d_m < 9$. Exceptions are the sets $\{1, 2, 3, 5, 7, 8\}$ and $\{1, 2, 3, 4, 6, 7, 8\}$. In the first case HOLE-MAX gives a corresponding score sequence $1^2, 2, 3^2, 5^2, 7, 8^2$, while in the second case algorithm HOLE-PAIRS presents the solution $1^2, 2, 3^2, 4, 6, 7, 8$.

If $d_m = 9$, then for the first three algorithms the critical sets are $\{2, 4, 5, 6, 7, 8, 9\}$, $\{1, 2, 5, 6, 7, 8\ 9\}$ and $\{1, 2, 4, 7, 8, 9\}$. HOLE-MAX solves these problems:

corresponding sequences are in the first case $2^3, 4^2, 5, 6, 7, 8, 9^2$, in the second case $1^2, 2, 5^2$, $6, 7, 8, 9^2$, and in the third case $1^2, 2^2, 4, 7^3, 8, 9^3$.

n	D	s	Algorithms	DIOPHANTINE
1	$\{3,5,6,7,8,9,10,11\}$	$4,2,1,1,1,1,2,3$	M + S	$5,1,1,2,1,1,1,1$
2	$\{3,4,5,6,8,9,10,11\}$	$4,1,1,1,2,1,2,3$	M + S	$5,1,1,1,1,1,2,1$
3	$\{3,4,5,6,7,10,11\}$	$4,1,1,1,1,6,1$	M + S	$4,2,1,1,2,1,1$
4	$\{2,4,5,6,7,8,9,10,11\}$	$3,2,1,1,1,1,1,2,2$	M + S	$4,1,1,1,1,1,1,2,1$
5	$\{2,3,5,9,10,11\}$	$3,1,2,5,3,1$	M + S	$1,3,5,1,1,1$
6	$\{2,3,5,7,9,10,11\}$	$3,1,2,2,2,1,4$	P + M + S	$3,2,2,2,1,1,1$
7	$\{2,3,4,8,9,11\}$	$3,1,1,4,1,6$	M + S	$3,1,3,2,2,1$
8	$\{2,3,4,5,6,8,9,10,11\}$	$3,1,1,1,1,2,1,2,2$	M + S	$3,1,2,1,1,1,1,1,1$
9	$\{2,3,4,5,6,7,8,10,11\}$	$3,1,1,1,1,1,1,3,2$	M + S	$3,1,1,1,2,1,1,1,1$
10	$\{2,3,4,5,6,7,8,9,11\}$	$3,1,1,1,1,1,1,2,3$	F	$3,1,1,1,1,2,1,1,1$
11	$\{1,7,11\}$	$3,2,14$	X	$3,1,3,2,1,1,1$
12	$\{1,4,5,7,9,10,11\}$	$2,3,1,3,1,2,2$	P + M + S	$3,1,3,2,1,1,1$
13	$\{1,3,5,6,7,10,11\}$	$2,2,2,1,1,3,4$	P + M + S	$3,1,3,2,1,1,1$
14	$\{1,3,4,5,8,10,11\}$	$2,2,1,1,4,2,2$	P + M + S	$2,2,2,1,3,1,1$
15	$\{1,2,6,8,9,11\}$	$2,1,4,3,1,4$	P + M + S	$2,2,4,2,1,1$
16	$\{1,2,5,9,10,11\}$	$2,1,3,4,1,5$	P + M + S	$2,2,3,4,1,1$
17	$\{1,2,4,8,10,11\}$	$2,1,2,5,2,3$	P + M + S	$1,2,4,3,1,1$
18	$\{1,2,4,8,9,11\}$	$2,1,2,5,1,4$	P + M + S	$1,2,4,2,2,1$
19	$\{1,2,4,7,9,10,11\}$	$2,1,2,4,1,2,2$	P + M + S	$1,3,2,3,1,1,1$
20	$\{1,2,4,6,9,10,11\}$	$2,1,3,1,3,2,2$	P + M + S	$2,1,2,4,1,1,1$
21	$\{1,2,4,5,7,10,11\}$	$2,1,2,1,2,3,4$	P + M + S	$2,1,2,1,4,1,1$
22	$\{1,2,3,7,8,9,10,11\}$	$2,1,1,4,1,1,2,3$	M + S	$2,1,2,3,1,2,1,1$
23	$\{1,2,3,6,10,11\}$	$2,1,1,3,7,1$	M + S	$1,1,2,6,1,1$
24	$\{1,2,3,5,8,10,11\}$	$2,1,1,2,4,2,2$	P + M + S	$1,2,1,4,1,1,1,1$
25	$\{1,2,3,5,7,11\}$	$2,1,1,2,2,7$	F	$1,2,1,1,6,1$
26	$\{1,2,3,5,7,10,11\}$	$2,1,1,2,2,3,4$	P + M + S	$2,1,1,2,2,3,4$
27	$\{1,2,3,5,7,8,11\}$	$2,1,1,2,2,1,6$	P + M + S	$2,1,1,2,2,3,1$
28	$\{1,2,3,5,6,9,11\}$	$2,1,1,2,1,3,5$	P + M + S	$2,1,1,3,1,3,1$
29	$\{1,2,3,5,6,8,11\}$	$2,1,1,2,1,2,6$	P + M + S	$2,1,1,2,1,4,1$
30	$\{1,2,3,4,9,10,11\}$	$2,1,1,1,5,2,4$	M + S	$1,1,2,3,4,1,1$
31	$\{1,2,3,4,5,9,10,11\}$	$2,1,1,1,1,4,2,3$	P + M	$2,1,1,1,2,4,1,1$
32	$\{1,2,3,4,5,6,7,8,11\}$	$2,1,1,1,1,1,1,3,4$	F	$1,1,2,1,1,1,1,3,1$

Table 3: Reconstruction results of the critical score sets ending with $d_m = 11$

If $d_m = 10$, then there are 18 sequences which are not reconstructable if we use only the algorithms BALANCING, SHORTENING, and SHIFTENING. Table 2

contains a possible reconstruction of these sequences. The used algorithms are P = Hole-Pairs, M = Hole-Max, S = Hole-Shift, X = Prefix-Shift, and F = Fit-All. The table contains also a shortest solution found by the brute force algorithm Diophantine described in [6]. Diophantine uses the algorithm described by Knuth [9, page 392].

If $d_m = 11$, then there are 72 sequences which are not reconstructable if we use only the algorithms Balancing, Shortening, and Shiftening. The majority of these sets can be reconstructed adding only P = Hole-Pairs and M = Hole-Max to the three basic algorithms. Table 3 is similar to the previous Table 2, but it contains only those examples (32 sets), whose reconstruction requires at least one of the algorithm Hole-Shift, Prefix-Shift, and Fit-All.

It is interesting to analyze the length of the critical sets. According to Theorem 7 for the length n of the score sequences corresponding to a score set $D = \{d_1, d_2, \ldots, d_m\}$ hold the bounds $\max(d_m + 1, 2d_1 + 2) \leq n \leq 2d_m$, and these bounds are sharp.

According to the lower bound in the case $d_m = 10$ we get $n \geq 11$. The data represented in Table 2 show, that Dophantine in 17 cases reaches this minimal length (the exception is $D = \{2, 3, 5, 7, 9, 10\}$) the approximate algorithms generate longer solutions in all cases.

If $d_m = 11$, then in the case of the critical sets the lower bound is $n \geq 12$. In the majority of cases Diophantine founds solutions of length 12, while the approximate algorithms never find a solution of length 12. The exact algorithm Diophantine in 29 cases found shorter sequence than the polynomial algorithms.

7 Summary

Checking all relevant score sets by polynomial time approximate algorithms we proved Theorem 3 for score sets whose maximal element is less than 12. Our proof is constructive, since we generated score sequences corresponding to the investigated score sets.

Acknowledgement

The author thanks the unknown referee for the proposed useful corrections, further PhD student of Eötvös Loránd University János Elek for making the computer experiments with algorithm Diophantine

References

[1] G. Chartrand, L. Lesniak, J. Roberts, Degree sets for digraphs, *Periodica Math. Hung.*, **7** (1976) 77–85. ⇒212

[2] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (third edition), The MIT Press/McGraw Hill, Cambridge/New York, 2009. ⇒216

[3] J. L. Gross, J. Yellen, P. Zhang, *Handbook of Graph Theory*, CRC Press, Boca Raton, FL, 2013. ⇒210

[4] M. Hager. On score sets for tournaments, *Discrete Math.*, **58** (1986) 25–34. ⇒210, 212

[5] A. Iványi, J. Elek, Degree sets of tournaments, *Studia Univ. Babeş-Bolyai, Informatica*, **59** (2014) 150–164. ⇒210, 215, 223, 226

[6] A. Iványi, L. Lucz, T. Matuszka, G. Gombos, Score sets in multitournaments, I. Mathematical results, *Annales Univ. Sci. Budapest., Rolando Eötvös Nom., Sectio Comp.*, **40** (2013) 307–320. ⇒210, 214, 215, 223, 226, 228

[7] A. Iványi, B. M. Phong, On the unicity of the score sets of multitournaments, in: *Fifth Conference on Mathematics and Computer Science* (Debrecen, June 9–12, 2004), University of Debrecen, 2006, 10 pages. ⇒213

[8] A. Iványi, S. Pirzada, Comparison based ranking, in: ed. A. Iványi, *Algorithms of Informatics*, Vol. 3, mondAt, Vác, 2013, 1209–1258. ⇒211

[9] D. E. Knuth, *The Art of Computer Programming*, Volume 4A. Addison Wesley, Upper Saddle River, NJ, 2011. ⇒228

[10] H. H. Landau, On dominance relations and the structure of animal societies. III., *Bull. Math. Biophysics*, **15** (1953) 143–148. ⇒211, 212

[11] Q. Li, Some results and problems in graph theory, *New York Academy of Science*, **576** (1989) 336–343. ⇒214

[12] V. Petrović, On bipartite score sets, *Zbornik Radova Prirodno-matematičkog Fakulteta Ser. Mat., Universitat u Novom Sadu*, **13** (1983) 297–303. ⇒213

[13] S. Pirzada, A. Iványi, M. A. Khan, Score sets and kings, in ed. A. Iványi, *Algorithms of Informatics*, mondAt, Vác, 2013, 1337–1389. ⇒214

[14] S. Pirzada, T. A. Naikoo, On score sets in tournaments, *Vietnam J. Math.*, **34** (2006) 157–161. ⇒212

[15] K. B. Reid, Score sets for tournaments, *Congressus Numer.*, **21** (1978) 607–618. ⇒210, 212

[16] K. B. Reid, Tournaments: Scores, kings, generalizations and special topics, *Congressus Numer.*, **115** (1996) 171–211. ⇒211, 212, 214

[17] K. Wayland, Bipartite score sets, *Canadian Math. Bull.*, **26** (1983) 273–279. ⇒213

[18] T. X. Yao, On Reid conjecture of score sets for tournaments. *Chinese Science Bull.*, **34** (1989) 804–808. ⇒210, 212, 213, 214

Received: June 28, 2014 • Revised: September 29, 2014